Содержание

1 Задание

Реализовать решение задачи аппроксимации функции вида:

$$f(\theta, x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

Функционалом качества является среднеквадратичная ошибка (MSE):

$$Q(\theta) = \frac{1}{l}\sum_{i=1}^{l}\left[f(\theta, x_i) - y_i\right]^2 = \frac{1}{l}\sum_{i=1}^{l} L(\theta, x_i, y_i) \to \min_{\theta \in R^n}$$

где, $\theta \in R^n$ – вектор подбираемых параметров
$f(\theta, x) = \theta_0 + \theta_1 x + \theta_2 x^2$ – аппроксимируемая функция
$(x_i, y_i)_{i=1}^{l}$ – обучающая выборка
$l = 10$
Целевая функция: MSE
Используемый язык: python 3.6.9
Реализуемые методы:

- Batch Gradient Descent – пакетный метод градиентного спуска
- Adadelta – метод скользящего среднего с адаптацией шага

1.1 Метод Batch Gradient Descent

Градиентный спуск — метод нахождения локального минимума или максимума функции с помощью движения вдоль градиента. Основная идея метода заключается в том, чтобы осуществлять оптимизацию в направлении наискорейшего спуска, а это направление задаётся антиградиентом.

На каждом шаге берем все объекты $(x_i, y_i) \in X^l$ из обучающей выборки и сразу обновляем вектор параметров θ по формуле:

$$\theta^{k+1} = \theta^k - \alpha_k \nabla_\theta Q(\theta^k) \tag{1}$$

В формуле (1) α_k – величина шага (*learning rate*), а $Q(\theta^k)$ – целевая функция, в нашем случае среднеквадратичная ошибка (Mean Squared Error). Проблема метода - необходимость для каждого улучшения параметров θ использовать всю выборку. Также необходимо подбирать скорость обучения для каждой задачи, потому что стандартное значение сильно влияет на скорость сходимости.

1.2 Метод Adadelta

Метод скользящего среднего с адаптацией шага или метод Adadelta. Идея метода заключается в том что происходит накопление информации о градиенте (поэлементные квадраты градиентов), а также накопление происходит по приращению (квадраты приращения). Так же вводится дополнительный член при вычислении приращения, который необходим для стабилизации. К преимуществам метода можно отнести отсутствие потребности подбора шага обучения (learning rate). Этот параметр никак не используется в процессе пересчета. Ниже представлен алгоритм метода:

Шаг 1. Необходимо задать параметры:
$\gamma \in (1, 0)$ – коэффициент сохранения, обычно принимается равным 0,9
$\varepsilon = 10^{-6} \div 10^{-8}$ – сглаживающий параметр
$\varepsilon_1 > 0$
$M^{-1} = 0$
$RMS[\Delta x]^{-1} > 0$
$RMS[\Delta x^2]^{-1} = 0$
$k = 0$

Шаг 2. Необходимо найти:

$$g^k = \nabla f^k(x^k)$$

$$G^k = g^k \odot g^k \tag{2}$$

$$M^k = \gamma M^{k-1} + (1 - \gamma)G^k \tag{3}$$

$$RMS[g]^k = \sqrt{M^k + \varepsilon} \tag{4}$$

$$\Delta x^k = g^k RMS[\Delta x]^{k-1} \oslash RMS[g]^k \tag{5}$$

$$D^k = \Delta x^k \odot \Delta x^k \tag{6}$$

$$RMS[\Delta x^2]^k = \gamma RMS[\Delta x^2]^{k-1} + (1 - \gamma)D^k$$

$$RMS[\Delta x]^k = \sqrt{RMS[\Delta x^2]^k + \varepsilon} \tag{7}$$

Шаг 3. Вычислить:

$$x^{k+1} = x^k - \Delta x^k$$

Шаг 4. Проверить выполнение условия:

$$\|x^{k+1} - x^k\| < \varepsilon_1 \tag{8}$$

Если условие (8) выполнено, то $x^* = x^{k+1}$. Иначе положить $k = k + 1$ и перейти к шагу 2.

В (3) происходит вычисление текущего момента M^k на основании предыдущего момента M^{k-1}, параметра γ, который является коэффициентом сохранения и квадрата градиента, который вычисляется в (2) путем поэлементного умножения градиента g^k на самого себя. Сохранение информации о приращении происходит при вычислении стабилизирующего члена (7), для вычисления которого используется квадрат приращения вычисляемый в (6) путем поэлементного умножения вектора на самого себя. Основная формула по которой вычисляется приращение это (5), в ней фигурируют градиент g^k и два RMS члена, один в числителе – $RMS[\Delta x]^{k-1}$, использующий информацию о приращениях и один в знаменателе – $RMS[g]^k$, использующий информацию о градиентах. Вычисления этих членов производится согласно (7) и (4) соответственно.

2 Аналитическое нахождение градиентов

Оба метода и **BGD** и **Adadelta** используют градиент функции в своих вычислениях. По этому необходимо привести аналитический метод для вычисления градиента функции потерь.

Функция потерь представляет собой:

$$L(\theta, x_i, y_i) = \sum_{i=1}^{l} [f(\theta, x_i) - y_i]^2$$

В свою очередь, для вычисления функции потерь необходимо указать функцию, которую мы аппроксимируем, эта функция имеет следующий вид:

$$f(\theta, x_i) = \theta_0 + \theta_1 x_i + \theta_2 x_i^2$$

В таком случае градиент потерь будет выглядеть вот так:

$$\nabla_\theta L(\theta, x_i, y_i) = \left(\frac{\partial L}{\partial \theta_0}, \frac{\partial L}{\partial \theta_1}, \frac{\partial L}{\partial \theta_2} \right)$$

$$\frac{\partial L}{\partial \theta_0} = 2 \sum_{i=1}^{l} [f(\theta, x_i) - y_i]$$

$$\frac{\partial L}{\partial \theta_1} = 2 \sum_{i=1}^{l} [f(\theta, x_i) - y_i] x_i$$

$$\frac{\partial L}{\partial \theta_2} = 2 \sum_{i=1}^{l} [f(\theta, x_i) - y_i] x_i^2$$

3 Код

Каждый из методов был реализован на языке python 3.6.9 и представляет собой функцию. Листинг каждой из функций представлен ниже:

3.1 Batch Gradient Descent

```
def BGD(theta, lr=.0002, steps=2000, tol=1e-4):
  loss_history = [float('inf'), loss(theta, x, y), ]
  for step in range(steps):
    theta -= lr * grad_loss(theta, x, y)
    loss_history.append(loss(theta, x, y))
  if abs(loss_history[step-1] - loss_history[step]) < tol:
    print(f'Tolerance exceeded, theta --> {theta}')
    del loss_history[0]
    break
  return theta, loss_history

```

Листинг 1: Batch Gradient Descent

3.2 Adadelta

```
def adadelta(theta, max_iterations=2000, gamma=0.9,
       tol=1e-4, e=1e-6):
  loss_history = [float('inf'), loss(theta, x, y)]
  dim = len(theta)
  E_g = np.zeros(dim)
  E_x = np.zeros(dim)
  for count in range(max_iterations):
    g = grad_loss(theta, x, y)
    E_g = (gamma * E_g) + ((1 - gamma) * g**2)
    RMS_E_x = RMS(E_x, e)
```

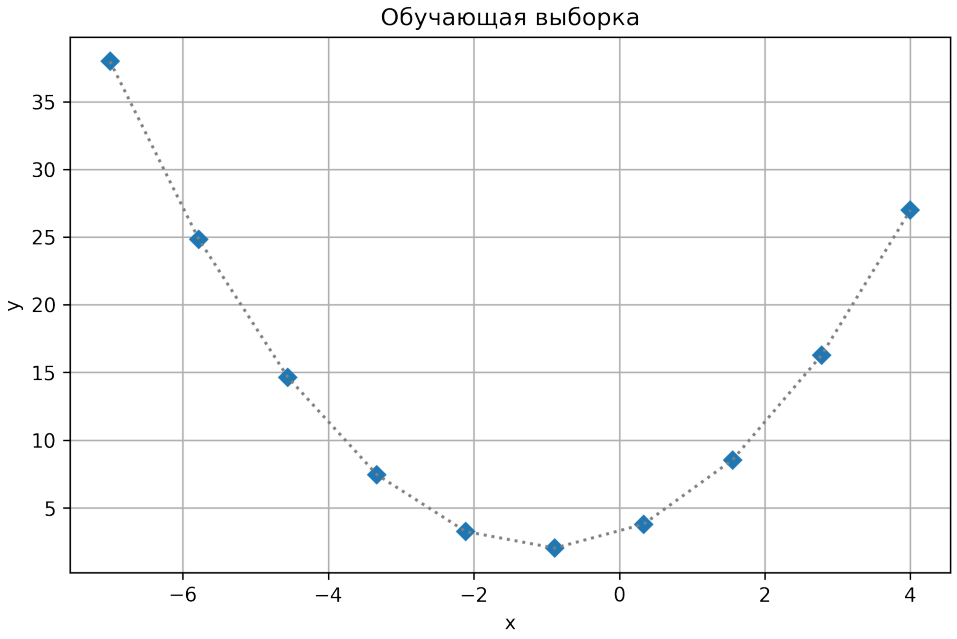
```
      RMS_E_g = RMS(E_g, e)
      delta_x = (RMS_E_x / RMS_E_g) * g
      E_x = (gamma * E_x) + ((1 - gamma) * delta_x**2)
      theta -= delta_x
      loss_history.append(loss(theta, x, y))
    if abs(loss_history[count-1] - loss_history[count]) < tol:
      print(f'Tolerance exceeded, theta --> {theta}')
      del loss_history[0]
      break
    return theta, loss_history

```

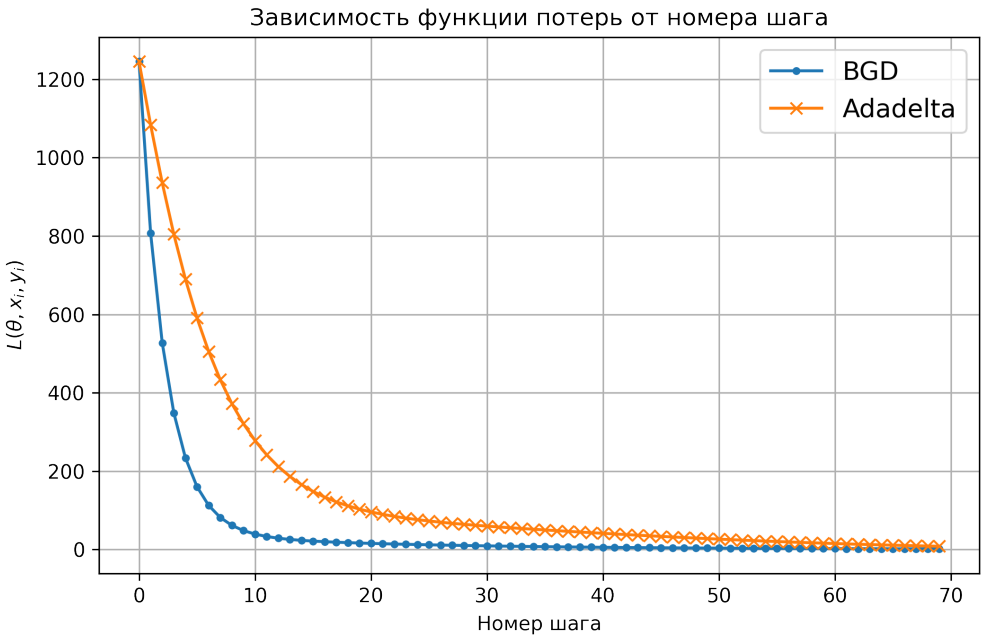
Листинг 2: Adadelta

4 Результаты

Для работы методов было сгенерировано обучающее множество из 10 точек на отрезке $[-7, 4]$. Был взят список значений x и передан в функцию $f(\theta, x) = \theta_0 + \theta_1 x + \theta_2 x^2$. Параметры θ были приняты $\theta_0 = 3, \theta_1 = 2, \theta_2 = 1$. Полученные точки представлены на графике ниже:



После генерации обучающего множества оно было передано в каждый из алгоритмов для решения задачи оптимизации – подбора таких параметров θ, которые минимизировали бы функционал качества $Q(\theta)$. График изменения значения функции потерь для каждого из методов представлен ниже:



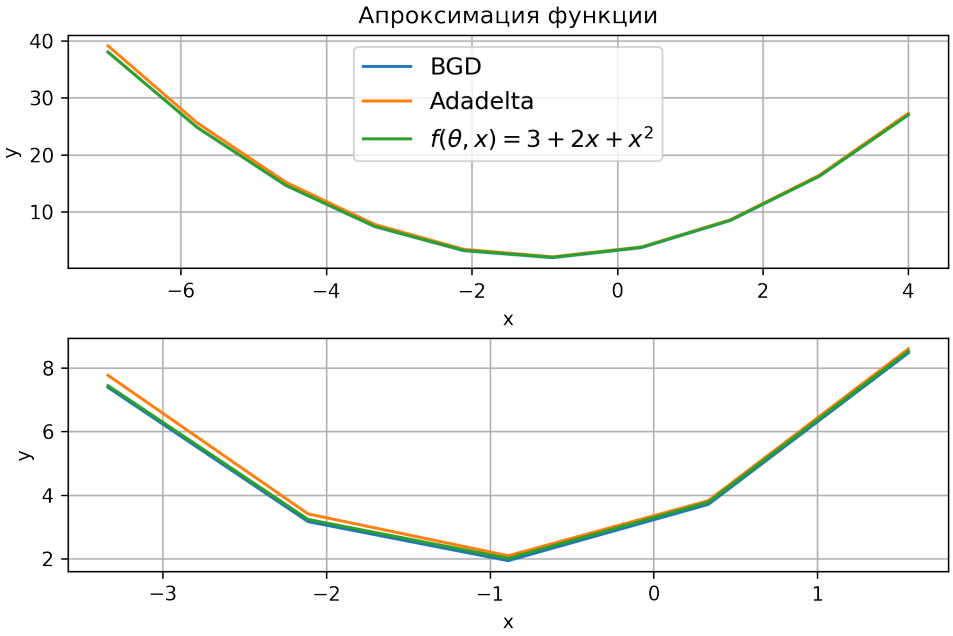
Параметры работы алгоритмов были следующие:

- **BGD**: $lr = .0002, steps = 2000, tol = 1e^{-4}$
- **Adadelta**: $max\_iterations = 2000, gamma = 0.9, tol = 1e^{-4}, e = 1e^{-4}$

Начальные значения для θ были одинаковыми для обоих методов и равны $\theta_0 = 2.5, \theta_1 = 1.3, \theta_2 = 0.4$.
В ходе работы методов были получены следующие значения параметров θ:

- **BGD**: $\theta_0 = 2.898, \theta_1 = 2.007, \theta_2 = 1.004$
- **Adadelta**: $\theta_0 = 3.050, \theta_1 = 1.983, \theta_2 = 1.019$

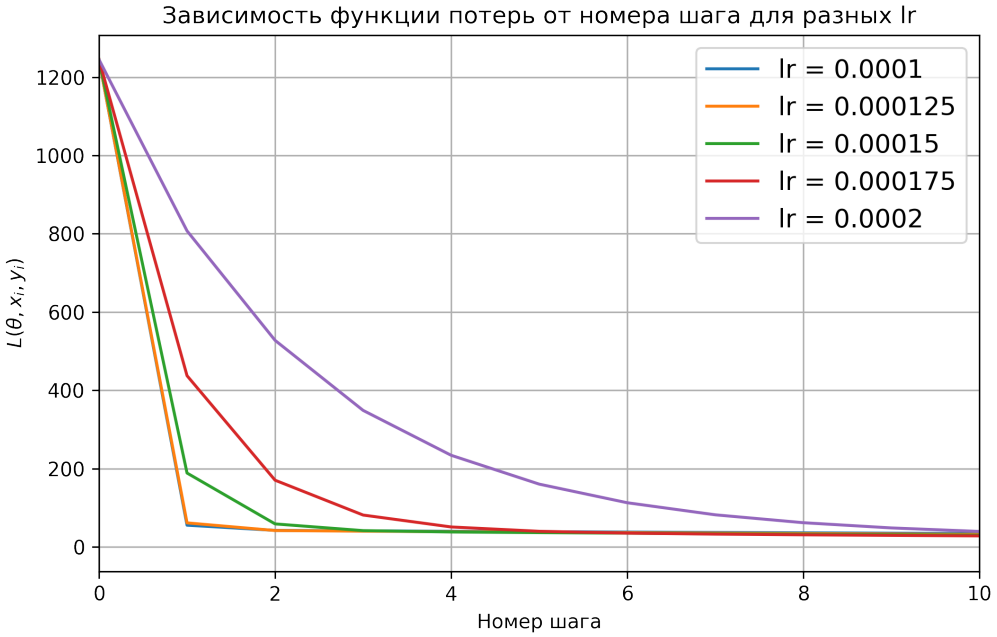
Аппроксимация исходной функции с полученными параметрами выглядит следующим образом:



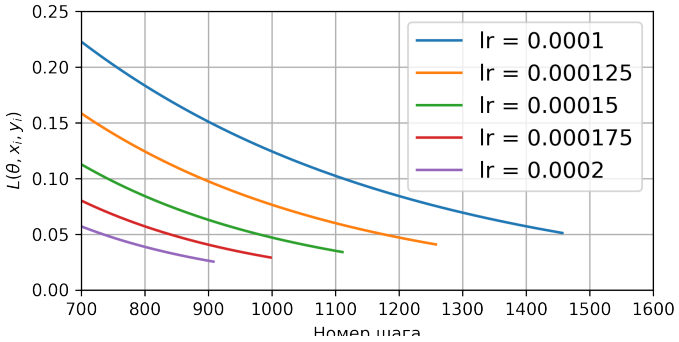
Верхний график отражает весь изначальный интервал, но из за наложения одной кривой на другую приведен нижний график, на котором заметны различия методов.

4.1 Анализ влияния параметров в методе BGD

Сначала проварьируем параметр learning rate для метода **BGD**. Было выбрано 5 значений параметра на отрезке $[0.0001, 0.0002]$. Ограничение на количество шагов оставалось низменным и было равно 2000. График изменения значения функции потерь от параметра lr и номера шага представлен ниже.



После 10 шага все кривые сливаются и график становится не информативным. Скорость сходимости метода увеличивается с увеличением скорости обучения, то есть первым сходится метод с $lr = 0.0002$, потом с $lr = 0.000175$. Это можно заметить на графике:



Полученные параметры θ сведены в таблицу:

learning rate	$\theta_0, \theta_1, \theta_2$
0.0001	2.898 2.007 1.004
0.000125	2.909 2.007 1.004
0.00015	2.917 2.006 1.004
0.000175	2.923 2.006 1.003
0.0002	2.928 2.005 1.003

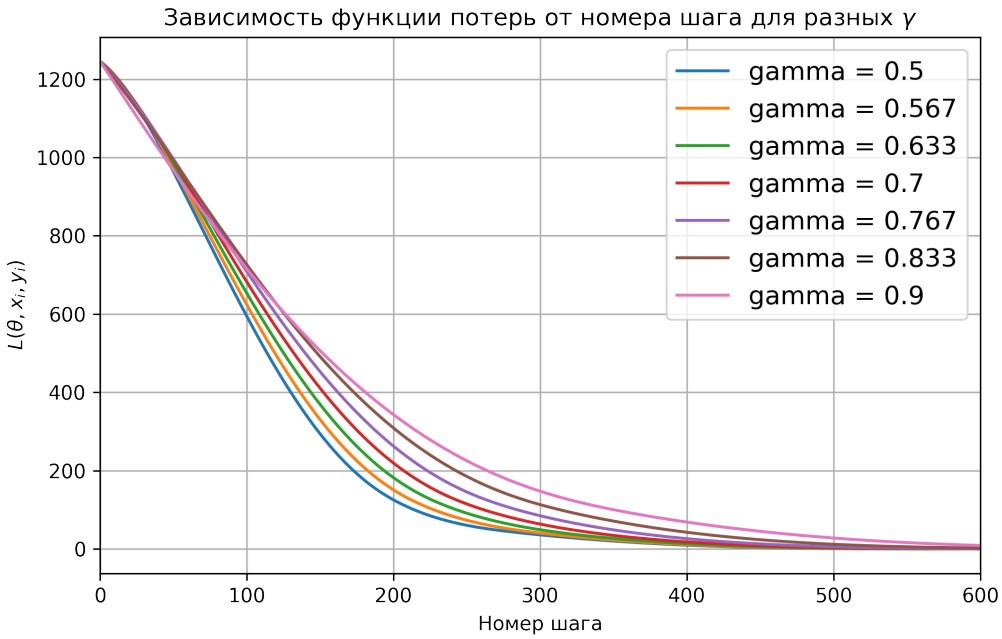
Как видно из графика, увеличение скорости обучения ведет к увеличению скорости сходимости. Варьирование же параметра отвечающего за максимальное количество итераций алгоритма не несет больших изменений в работе метода потому что алгоритм, при решении данной задачи при использовании $lr = 0.0002$ сходится менее чем за 1000 шагов. Это означает что при количестве шагов $\ll 1000$ возможна потеря точности.

4.2 Анализ влияния параметров в методе Adadelta

Далее перейдем к варьированию параметров метода **Adadelta**. Всего в этом методе используется 4 параметра,а именно:

- $\gamma \in (0, 1)$ – коэффициент сохранения
- ε – сглаживающий параметр
- ε_1 – параметр ответственный за раннюю остановку алгоритма
- max_steps – максимальное число итераций. Оставим его равным 2000 как и в **BGD**

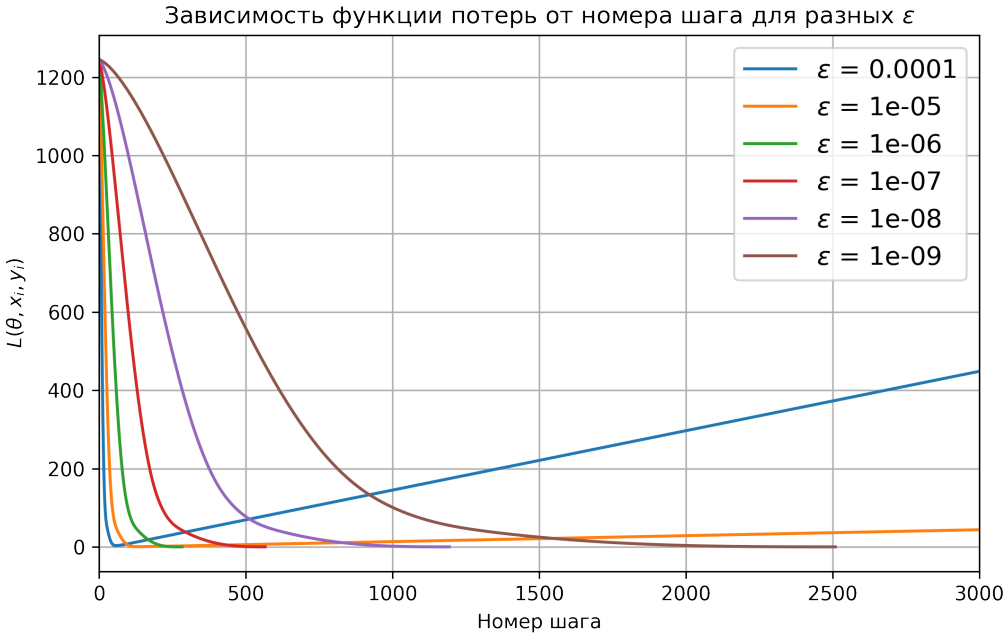
Сначала проварьируем параметр γ. Выделим 8 значений на интервале $(0.5, 0.9)$. Результаты изменения функции потерь по шагам отобразим на графике.



На графике, после 600 шага все кривые опять сливаются в одну, но по выводу метода можно заметить что при увеличении параметра γ скорость сходимости метода падает. Если положить параметр γ почти равным единице, то это может привести к параличу алгоритма.Посмотрим как варьирование γ сказалось на вычислении θ. Результаты представлены в таблице:

γ	$\theta_0, \theta_1, \theta_2$
0.5	3.022 1.995 0.998
0.557	3.025 1.994 0.998
0.614	3.031 1.992 0.997
0.671	3.037 1.991 0.997
0.729	3.046 1.991 0.996
0.786	3.05 1.992 0.995
0.843	3.041 1.992 0.995
0.9	3.027 1.991 0.995

Следующий параметр который будет проварьирован, это ε. Выделим 6 значений, а именно: $1e^{-4}, 1e^{-5}, 1e^{-6}, 1e^{-7}, 1e^{-8}, 1e^{-9}$. Параметр γ при этом менять не будет и отставим его равным 0.5 как лучший из возможных. Построим график изменения функции потерь.



Из приведенного графика можно заметить что первые два параметра не дают графику сойтись. Однако следующие значения позволяют методу сходиться довольно быстро и чем больше становится параметр ε тем медленнее скорость сходимости. Надо отметить что при использовании параметра $\varepsilon = 1e^{-9}$ методу не удается достигнуть заданной точности за 2000 шагов, но при их увеличении метод сходится примерно на шаге номер 2500. Полученные значения параметров θ при варьировании сведены в таблицу:

ε	$\theta_0, \theta_1, \theta_2$
$1e^{-4}$	NaN
$1e^{-5}$	NaN
$1e^{-6}$	3.064 1.989 0.999
$1e^{-7}$	3.022 1.995 0.998
$1e^{-8}$	3.008 1.998 0.999
$1e^{-9}$	3.007 1.996 0.999

5 Вывод

В ходе выполнения курсовой работы, были исследованы два метода оптимизации: пакетный метод градиентного спуска и метод скользящего среднего с адаптацией шага. Была написана реализация каждого метода на python. Необходимо отметить что пакетный метод градиентного спуска требует меньших вычислительных ресурсов и работает быстрее. В то время как метод скользящего среднего с адаптацией шага позволяет совершать более гибкую адаптацию, что несет в себе необходимость использования больших вычислительных ресурсов и мощностей. Однако при использовании нулевых или случайных значений метод adadelta сходился быстрее градиентного спуска.

Список литературы

[1] Павел Садовников. *Методы оптимизации нейронных сетей.* Habr.com, URL: https://habr.com/ru/post/318970/, 2017.

[2] Matthew D. Zeiler. *ADADELTA: AN ADAPTIVE LEARNING RATE METHOD.* Arxiv.org, URL: https://arxiv.org/pdf/1212.5701.pdf, 2012.

[3] Sebastian Ruder *An overview of gradient descent optimization algorithms.* Ruder.io, URL: https://ruder.io/optimizing-gradient-descent/, 2016.

[4] Пантелеев А.В., Летова Т.А. *Методы оптимизации в примерах и задачах.*, 2005.